16824 HW2 Report

Jagjeet Singh (jagjeets)

Notes about submission:

1. I had my ***data*** folder inside ***code***. Since the size was too large, I had to remove the data. To run the code, please place the *data* folder in ***hw2/code/***
2. The tensorboard file for the entire training was too huge (3.5GB). As such, I have submitted the logs only for the last epoch.

Task 0

0.1 What classes does the image at index 2018 contain?

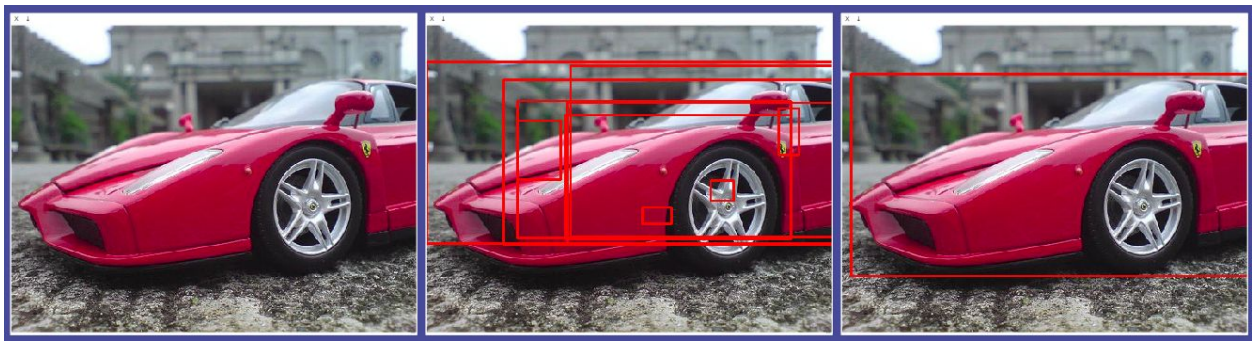
The image at index 2018 contains 'Car' class

0.2: What is the filename of the image at index 2018?

/home/ubuntu/hw2/hw2/code/data/VOCdevkit2007/VOC2007/JPEGImages/003998.jpg

Q 0.3 Use visdom+cv2 to visualize the top 10 bounding box proposals for image at index 2018. You would need to first plot the image and then plot the rectangles for each bounding box proposal.

Q 0.4 Use visdom+cv2 to visualize the ground truth boxes for image at index 2018.



Original Image | Image with Proposed BB | Image with Ground Truth BB

Task 1

1.1 Describe functionality of the completed TODO blocks

custom.py

def find_classes(): Used to obtain the list of class names and index corresponding to each

def make_dataset(): Used to generate the training data (image paths and corresponding annotations) from imdb object and class to index mapping
class LocalizerAlexNet(): Defines the architecture of the network. __init__ function initializes (and defines) the layers and forward() function is used to define the sequential flow of forward prop.

def localizer_alexnet(): Used to initialize the LocalizerAlexNet model. If pretrained is True, then features are initialized from pretrained AlexNet and classifier from xavier initialization. If pretrained is False, all the layers are Xavier Initialized.
def IMDBDataset(): Defines a custom Dataset class for Dataloader. Each call recieves an *index* number and returns the data in the format which can be directly used for Training or Validation. Images and Target labels (binary array of size batchx20) are returned

main.py

def main():

- The loss function is declared as the criterion. Since we are dealing with multilabel classification, Binary Cross Entropy loss is used. size_average=True because learning rate is taking the factor of 20 into account.
- Stochastic Gradient Descent is declared with specified momentum and weight decay
- Loggers are created for Visdom and TensorBoard. Since visdom requires AWS DNS to initialize with a server address, get_DNS() function is created to automatically fetch the DNS name.

def train():

- Output from the model is taken. Since the output is 20x20, MaxPool is taken to get image level labels and then sigmoid is applied to get probabilities. Finally, loss.backward() is called to backpropagate.
- Visualization is done as asked in the assignment. 4 batches per epoch are used and 4 images per batch visualized. For visualizing heatmaps, output scores of the model are resized followed by some preprocessing and then sent to visdom and tensorboard as images.

def validate(): Similar to train(). The only difference is that the loss is not backpropagated.
def metric1(): mean Average Precision is calculated here

1.2 What is the output resolution of the model?

The output resolution is 29x29 for each of the 20 channels

1.4 In the first few iterations, you should observe a steep drop in the loss value. Why does this happen?

This happens because the model tries to achieve the easy task and predicts everything as 0. Since the ground truth data is highly imbalanced with very few 1s, this prediction decreases the the loss sharply.

Precision = true positives / (true positives + false positives).

In the case where everything is predicted as False, the false positives are very low. As such, Precision will give a deceptively good value and mAP will not be a good metric to evaluate.

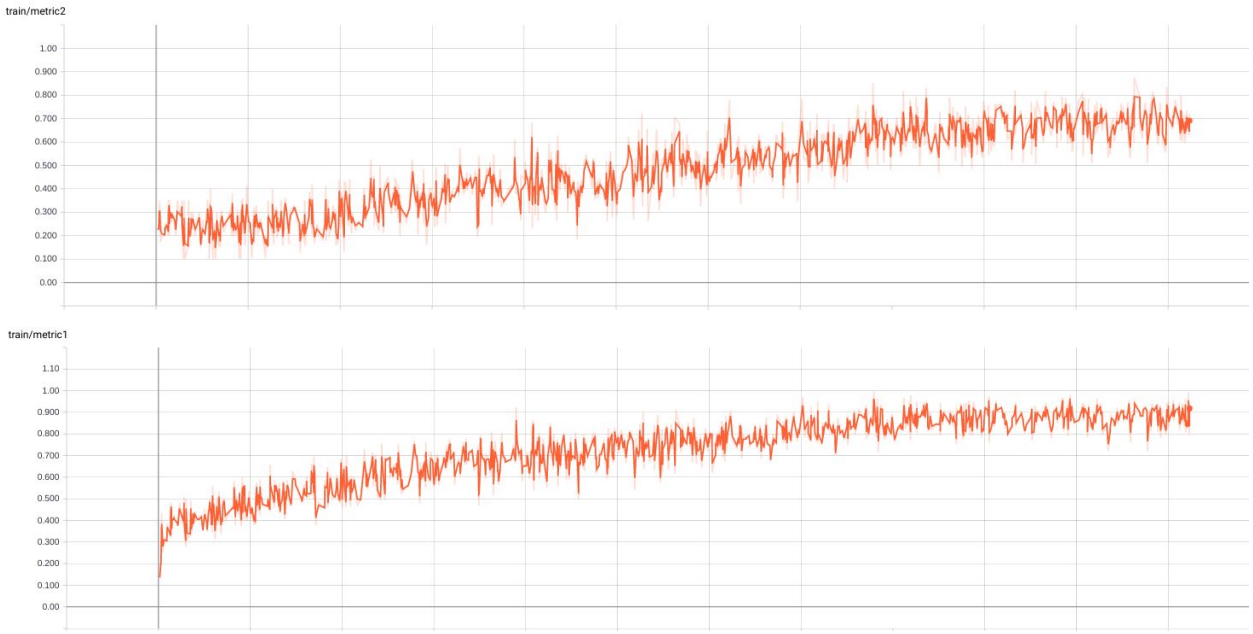
1.5 Even though there is a steep drop in loss in the first few iterations `metric2` should remain almost constant. Can you name one such metric? Describe functionality of the completed TODO blocks

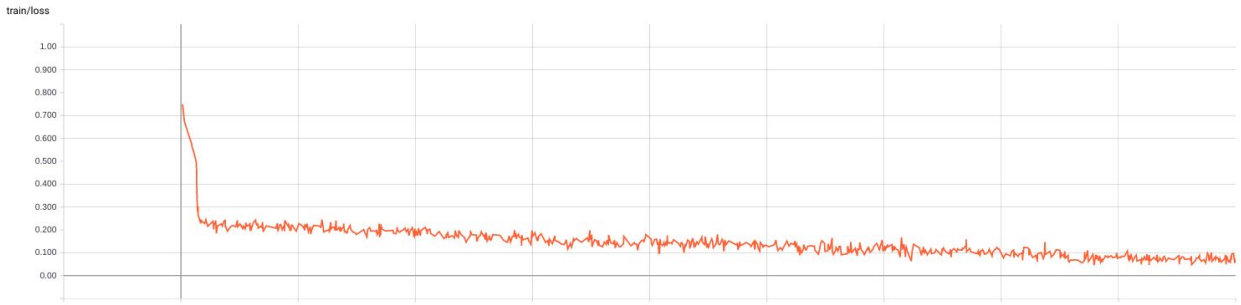
The explanation above suggests that Precision is not a good approach.
The next approach to think about is recall as it gives high penalty if there are false negatives. I tried average recall score but that also was not stable.
Next thought is to have both Precision and recall and also have high weightage to recall. As such, I eventually used F-beta score with beta=2. Therefor, both precision and recall were accounted but recall was given twice the weightage.

main.py:

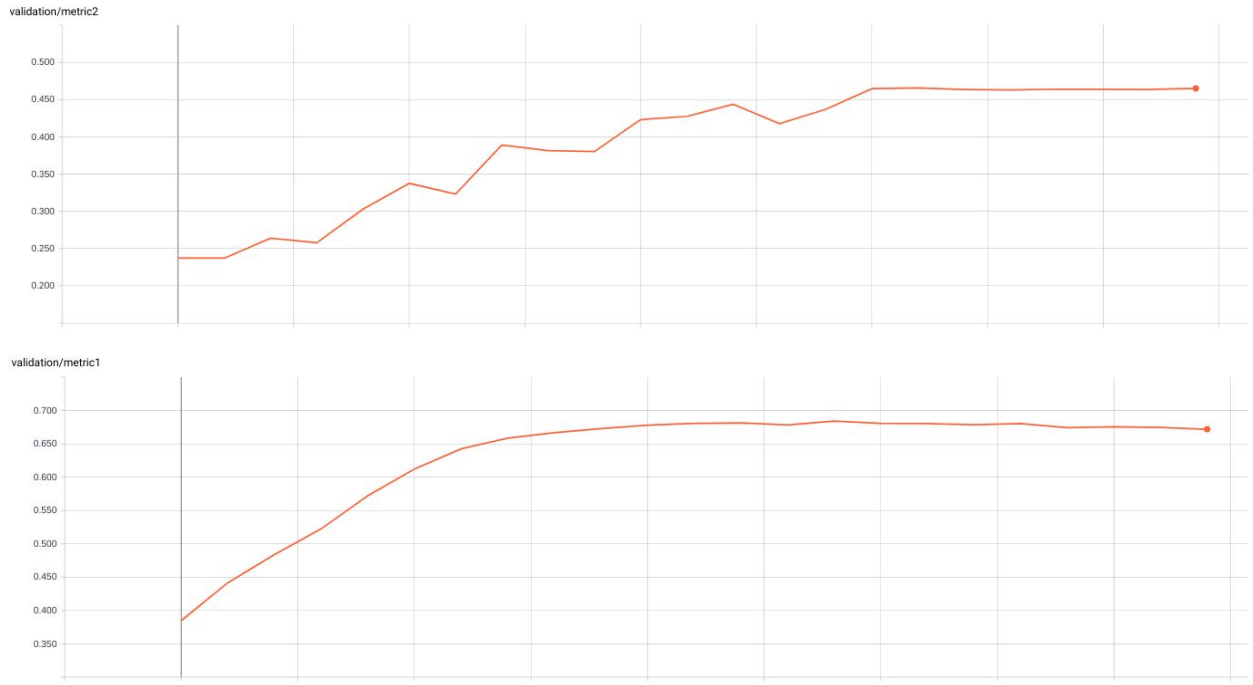
def metric2(): f-beta score is calculated here with threshold of 0.5 and beta value 2.

1.5.1 screenshot of tensor board metric1, metric2 (and loss) on the training set





1.5.2 Screenshot of tensor board metric1, metric2 on the validation set



1.5.3 Screenshot of tensor board showing images and heat maps for the first logged epoch

Observations: The heatmap was very random. It was difficult to see high score for any class in the image

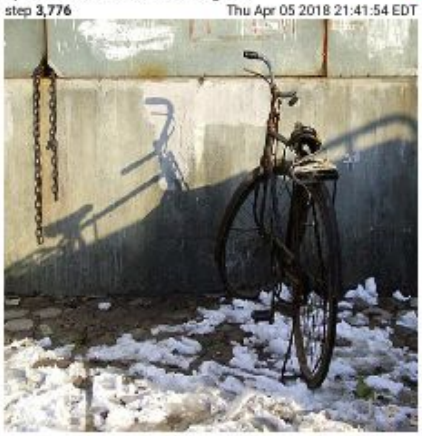
(Images on the next page)

ep0_iter3776_batch117_image2

ep0_iter3776_batch117_image2/0
step **3,776** Thu Apr 05 2018 21:41:55 EDT

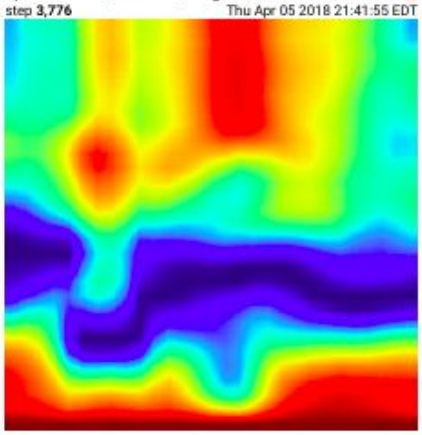
ep0_iter3776_batch117_image1

ep0_iter3776_batch117_image1/0
step **3,776** Thu Apr 05 2018 21:41:54 EDT



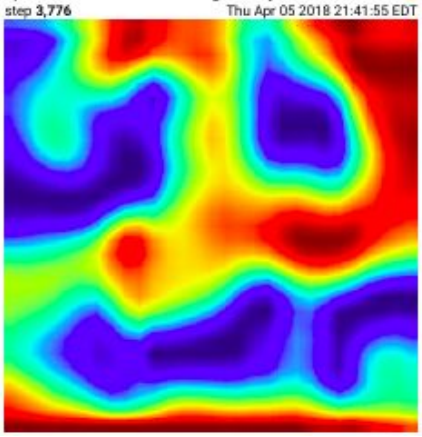
ep0_iter3776_batch117_image2_car

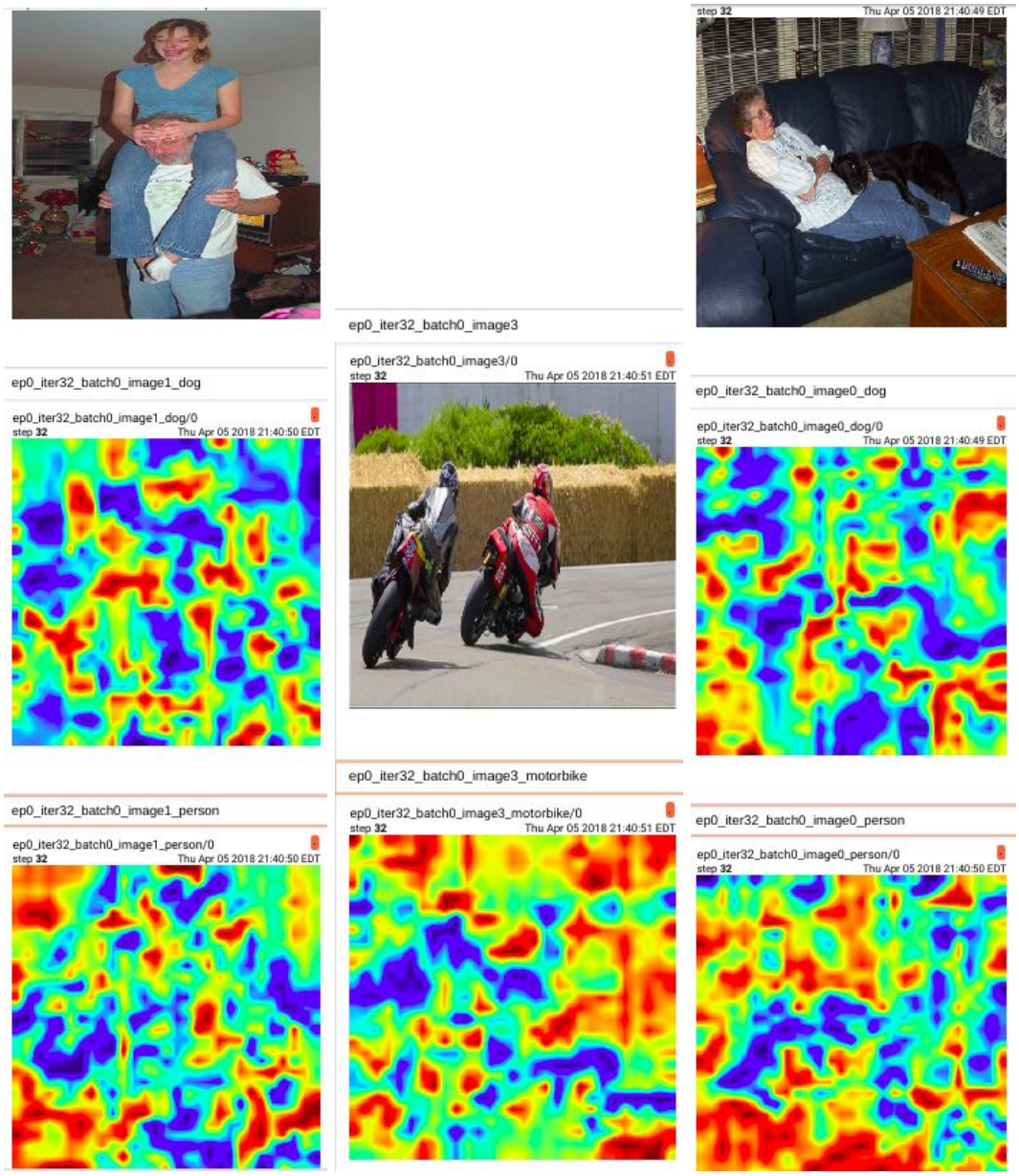
ep0_iter3776_batch117_image2_car/0
step **3,776** Thu Apr 05 2018 21:41:55 EDT



ep0_iter3776_batch117_image1_bicycle

ep0_iter3776_batch117_image1_bicycle/0
step **3,776** Thu Apr 05 2018 21:41:55 EDT





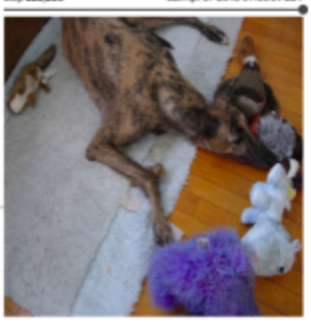
1.5.3 Screenshot of tensor board showing images and heat maps for the last logged epoch

Observations: The heatmap was better than the previous case but still the quality was poor. Also, in most of my heatmaps, **I have some noise on the borders which results in red color on the sides.** It might be because of colormaps that I used.

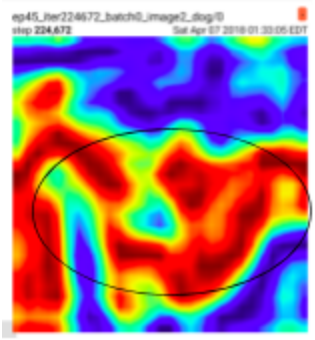
ep45_iter224672_batch0_image2

ep45_iter225280_batch19_image0

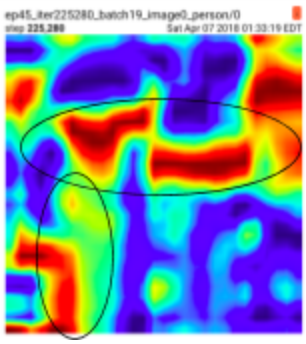
ep45_iter225280_batch19_image3/0
step 225,280
Sat Apr 07 2018 01:55:59 EDT



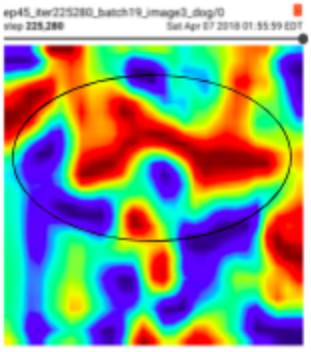
ep45_iter224672_batch0_image2_dog



ep45_iter225280_batch19_image0_person

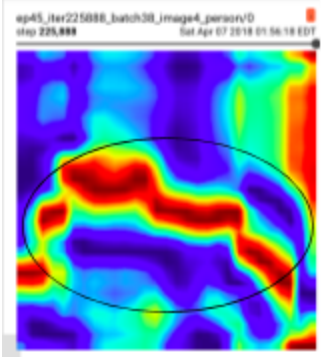


ep45_iter225280_batch19_image3_dog

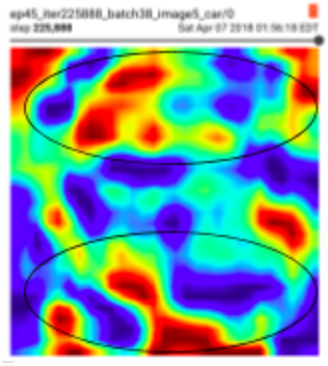


ep45_iter225888_batch38_image4

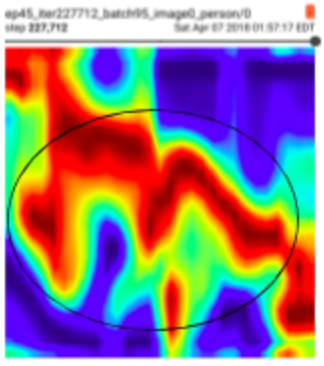
ep45_iter225888_batch38_image4_person



ep45_iter225888_batch38_image5_car

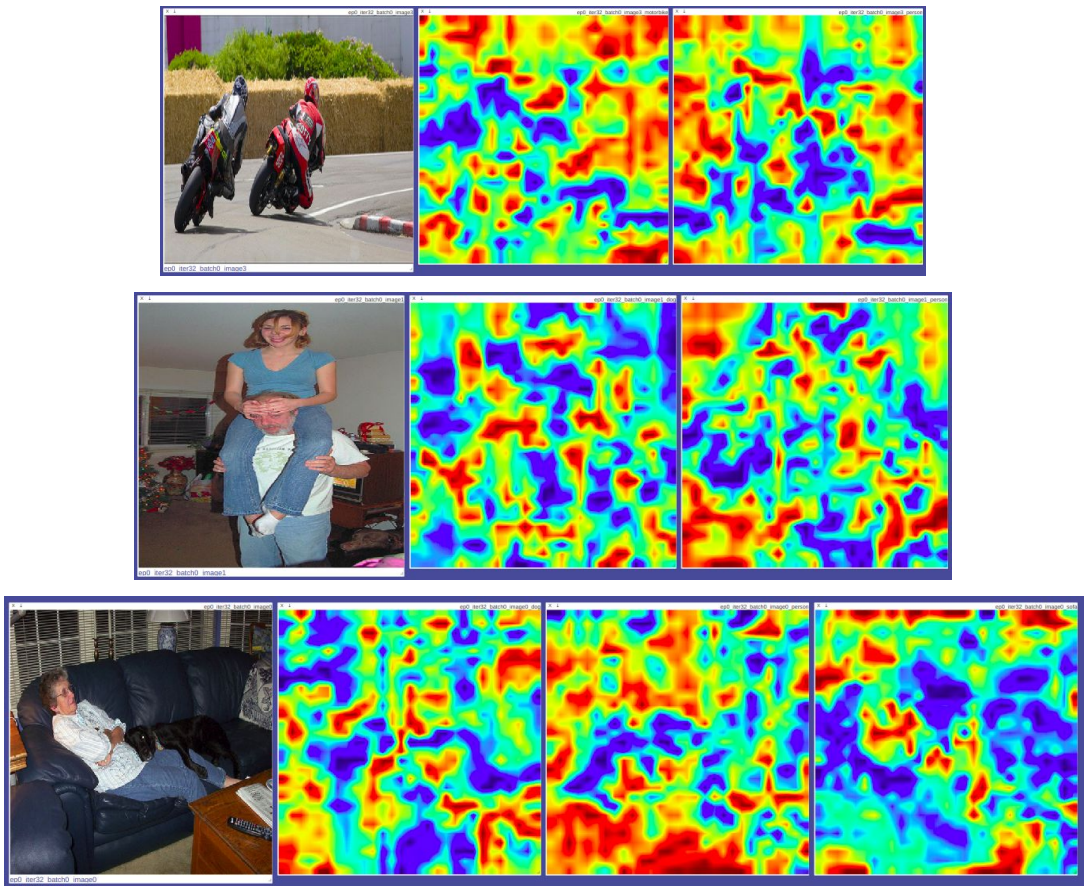


ep45_iter227712_batch95_image0_person



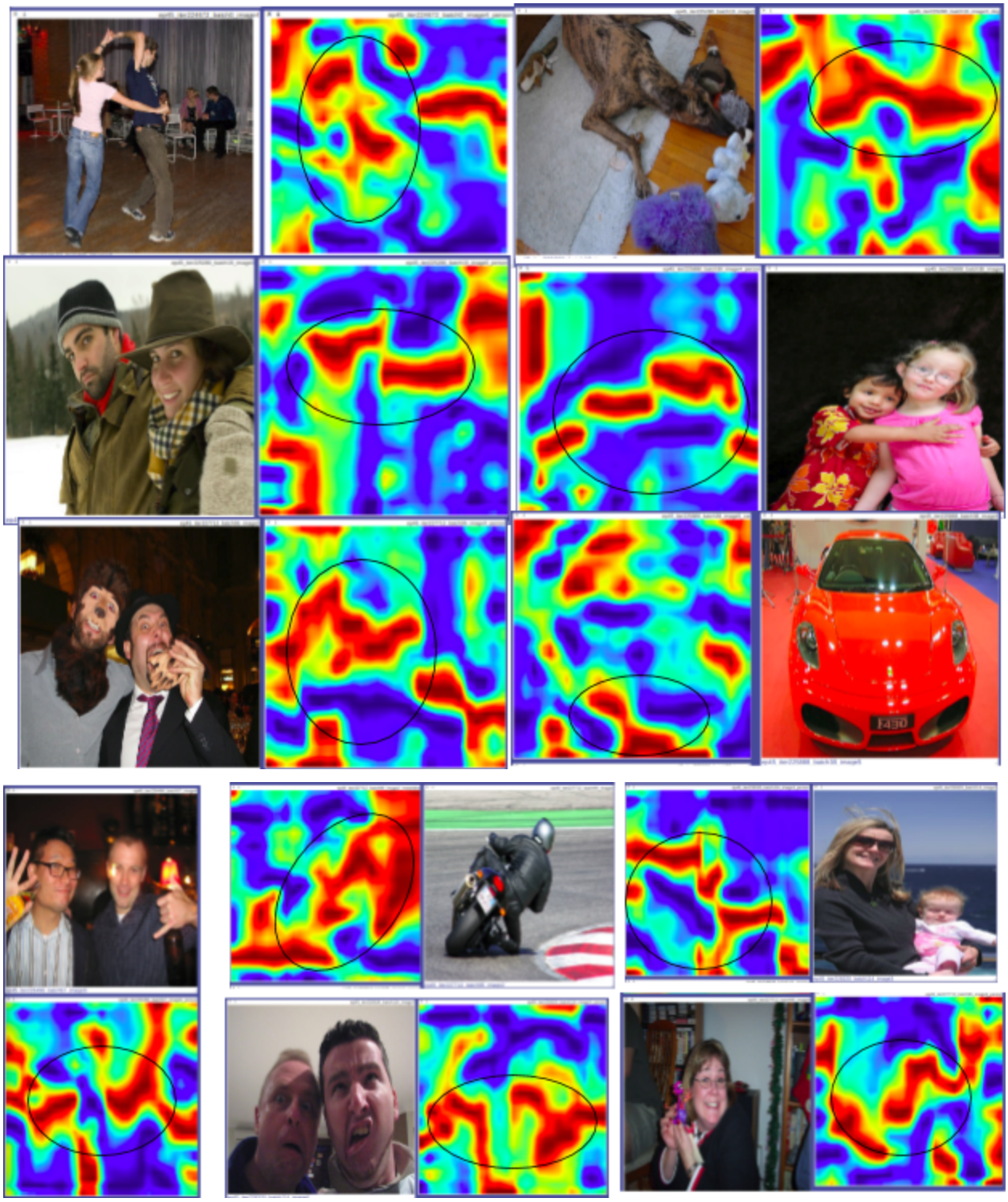
1.5.4 Use visdom filter in the browser to search for the first logged epoch

Observations: The heatmap is exactly the same as tensorboard. I didn't use vis.heatmap. I instead preprocessed the heatmap and plotted as an image. As such, both visdom and tensorboard are showing the same thing.



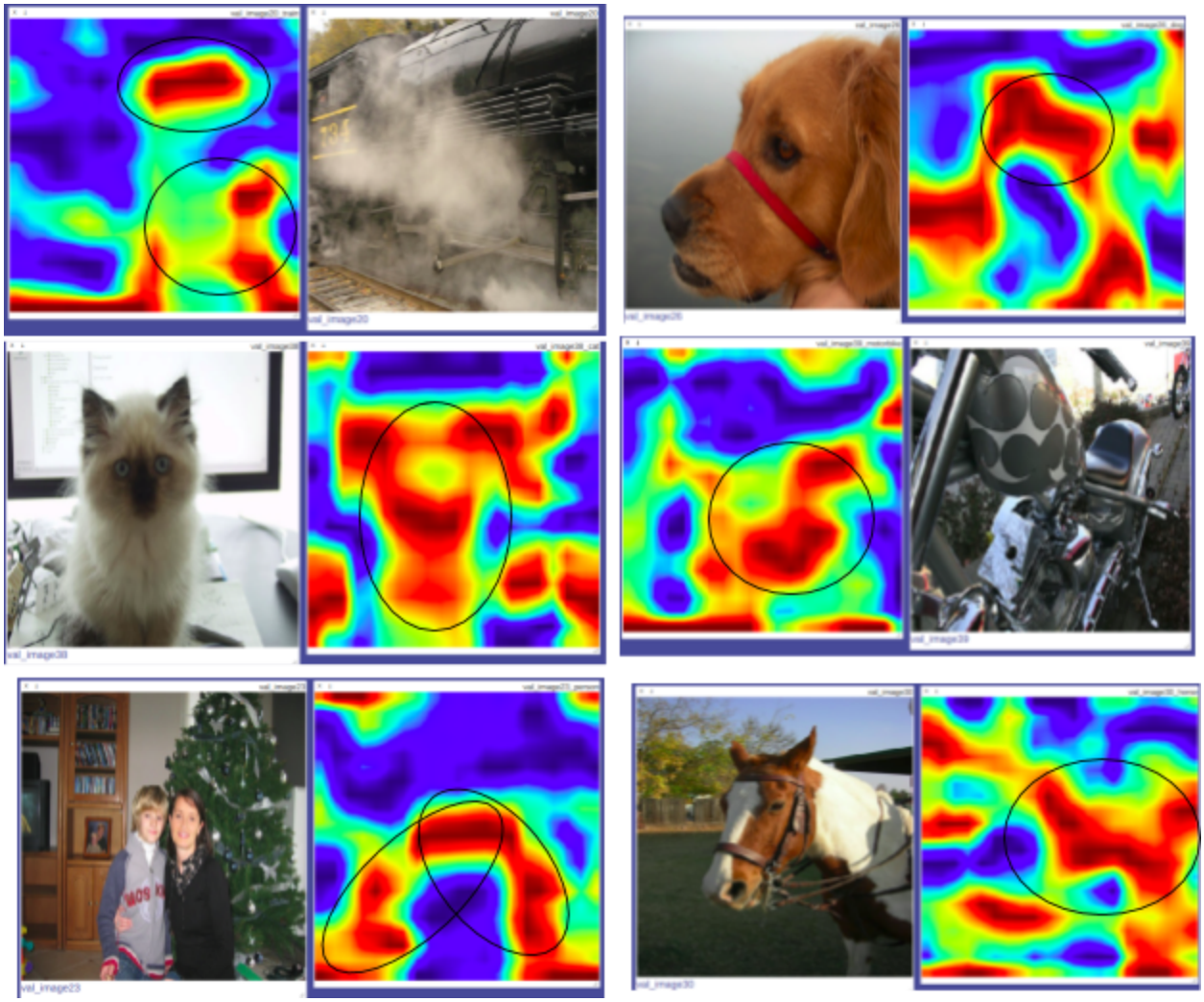
1.5.5 Use visdom filter in the browser to search for the last logged epoch, include screenshot

Observation: Heatmaps worked really well for localizing salient features of ‘Humans’, but not so much for other classes



1.5.6 Visdom screenshot for 20 randomly chosen validation images and heat maps

Some sample (<20) images are shown below. Unlike the last training epoch, the heatmaps in the validation set showed good results for classes other than humans as well.



1.5.7 Report training loss, validation metric1, validation metric2 at the end of training

Final value for Validation metric1 is **0.6719**
Final value for Validation metric2 is **0.4649**
Final value for Training Loss **0.0698**

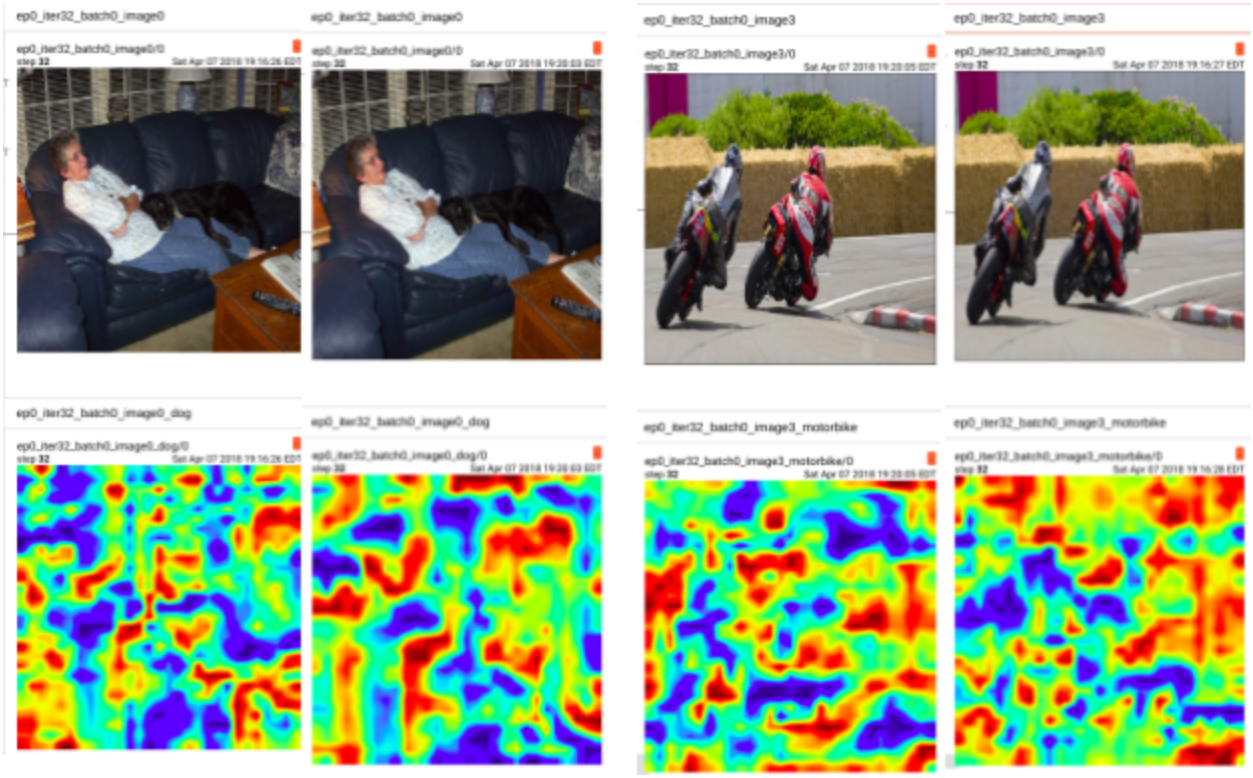
1.6 Describe functionality of the completed TODO blocks

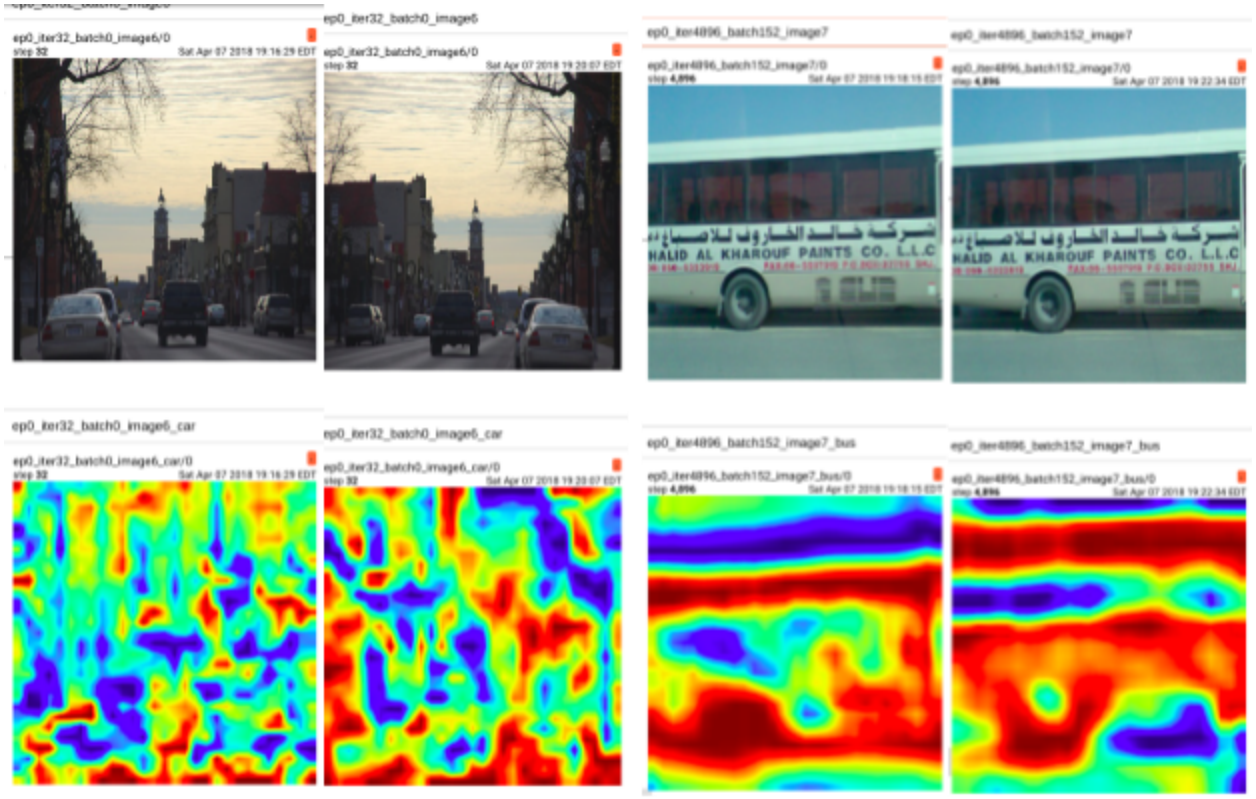
custom.py

class LocalizerAlexNetRobust(): Defines the architecture of the network. __init__ function initializes (and defines) the layers and forward() function is used to define the sequential flow of forward prop. The only difference is the addition of dropout before pooling the model outputs to randomly suppress some maximum values.
def localizer_alexnet_robust(): Used to initialize the LocalizerAlexNet model. If pretrained is True, then features are initialized from pretrained AlexNet and classifier from xavier initialization. If pretrained is False, all the layers are Xavier Initialized.

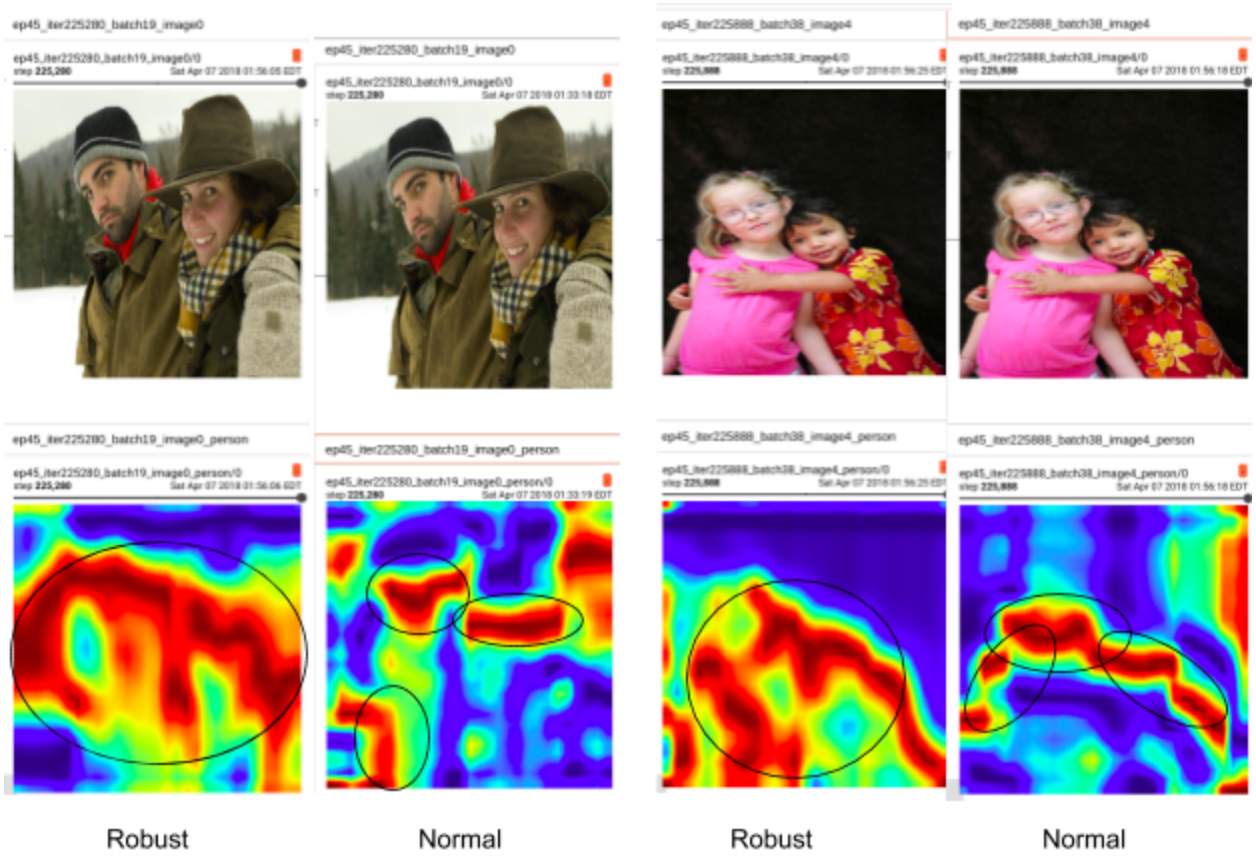
1.6.1 Screenshot of tensor board showing images and heat maps for the first logged epoch

(Images on the next page)

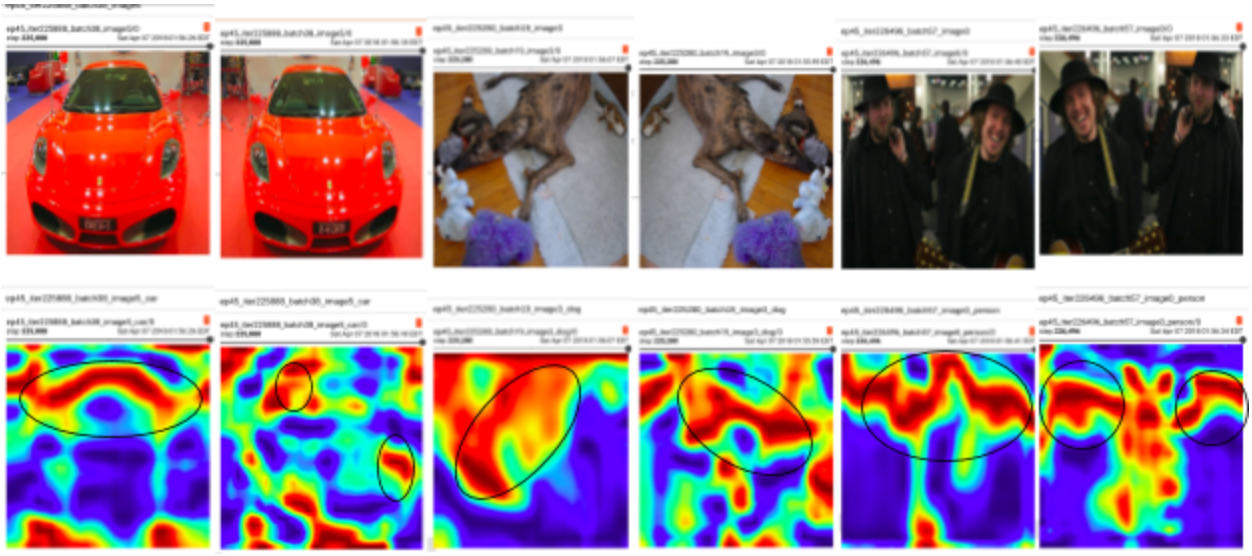



ep0_iter32_batch0_image6/0
ep0_iter32_batch0_image6/0
ep0_iter4896_batch152_image7
ep0_iter4896_batch152_image7
HALID AL KHAROUF PAINTS CO. L.L.C
HALID AL KHAROUF PAINTS CO. L.L.C
ep0_iter32_batch0_image6_car
ep0_iter32_batch0_image6_car
ep0_iter4896_batch152_image7_bus
ep0_iter4896_batch152_image7_bus
Normal
Robust
Normal
Robust

1.6.2. Screenshot of tensor board showing images and heat maps for the last logged epoch



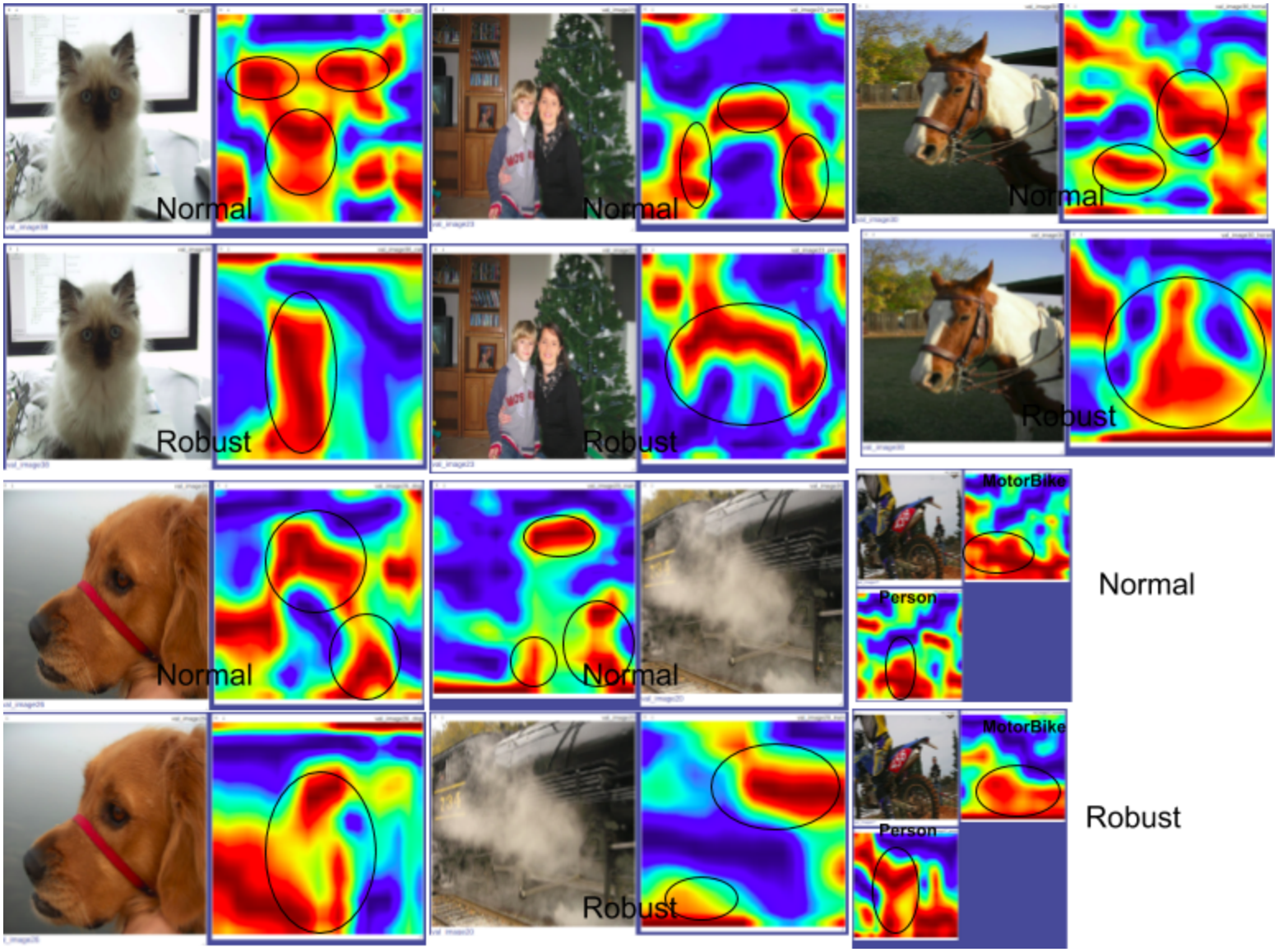
LocalizerAlexNetRobust heatmap shows high values for majority of the human body but LocalizerAlexNet heatmap shows high values only for disjoint salient features



Robust Normal Robust Normal Robust Normal

Robust is smooth and encompasses majority of the object. Also, some maximal values which were present in LocalizerAlexNet are suppressed.

1.6.3. Visdom screenshot for 20 randomly chosen validation images and heat maps

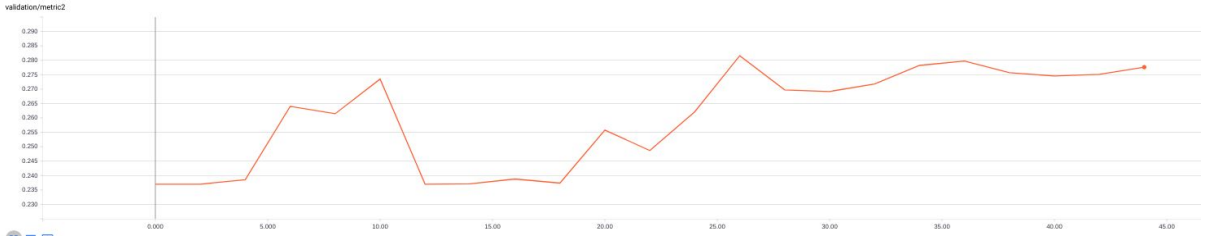


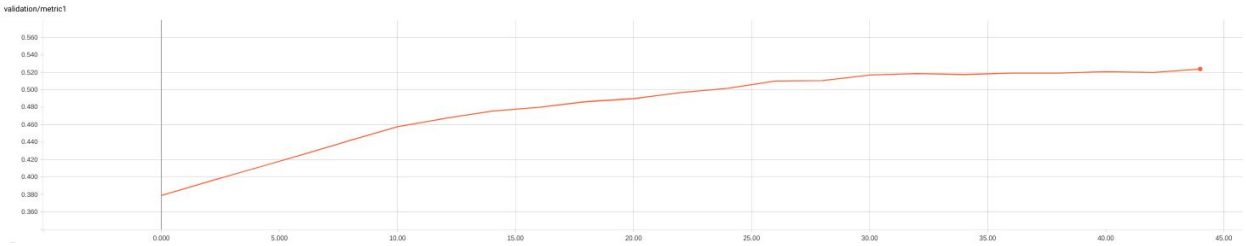
1.6.3. Report training loss, validation metric1, validation metric2 at the end of training

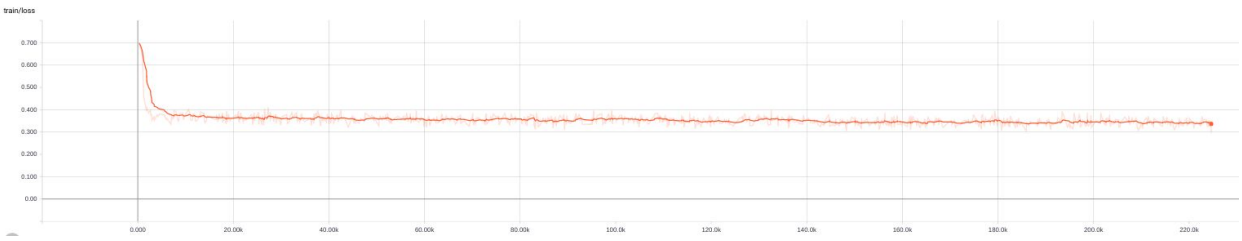
Final (after 45 epochs) value for Validation metric1 is **0.523**

Final (after 45 epochs) value for Validation metric2 is **0.277**

Final (after 45 epochs) value for Training loss is **0.326**



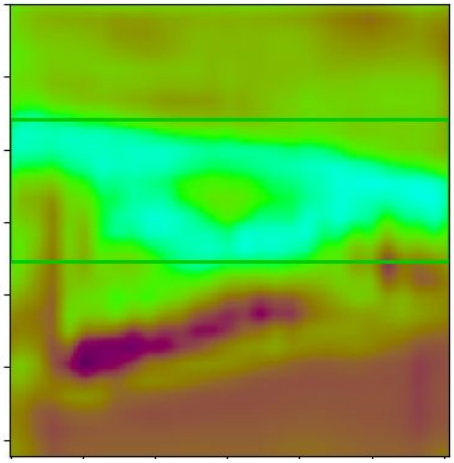
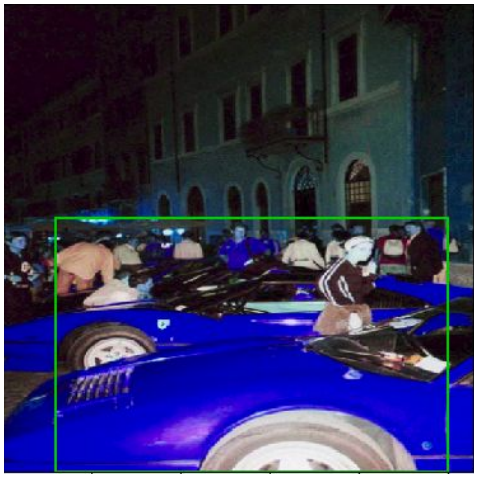
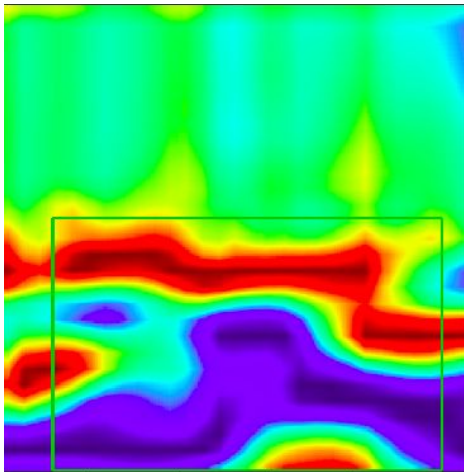
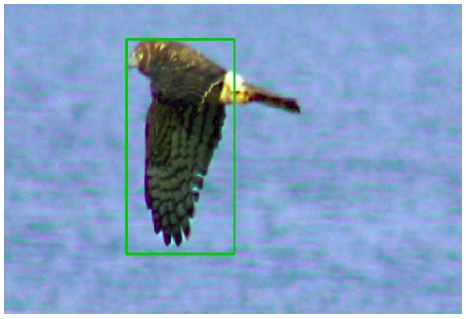
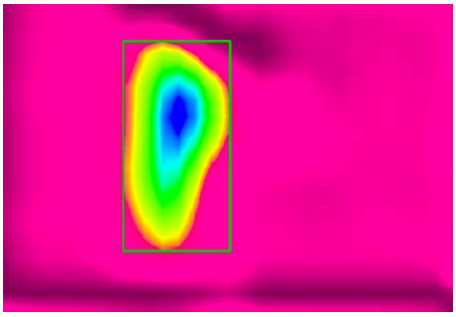




1.7. The outputs of the model from Q1.6 are score maps (or heat maps). Try to come up with a reasonable algorithm to predict a bounding box from the heatmap

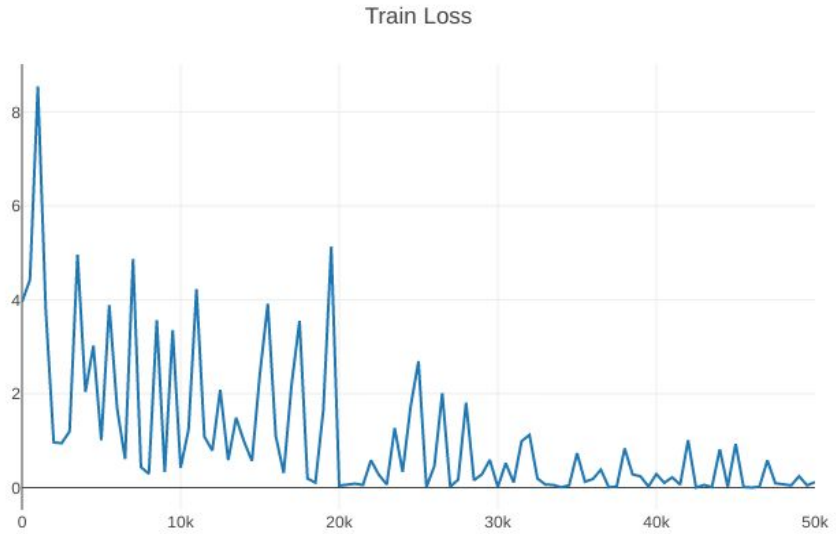
Note: Following are some sample images. The approach worked for simple heatmaps but failed when applied to those where multiple disconnected components correspond to the same class. Also, as explained before, because of noise at the boundaries, it was difficult to predict the bounding boxes. The task of comparison with ground truth bounding boxes and evaluating mAP hasn't been made

Approach: Find min x, max x, min y and max y that have high energy in the heatmap. Threshold was set to 0.9

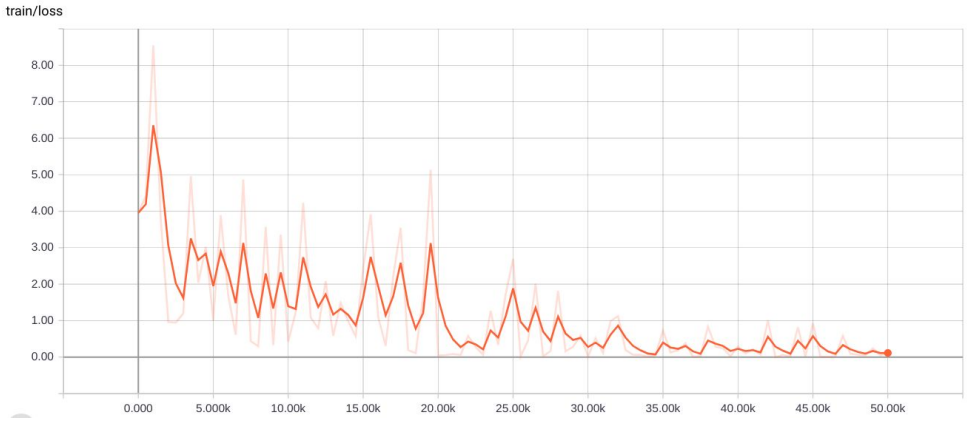







Task 2

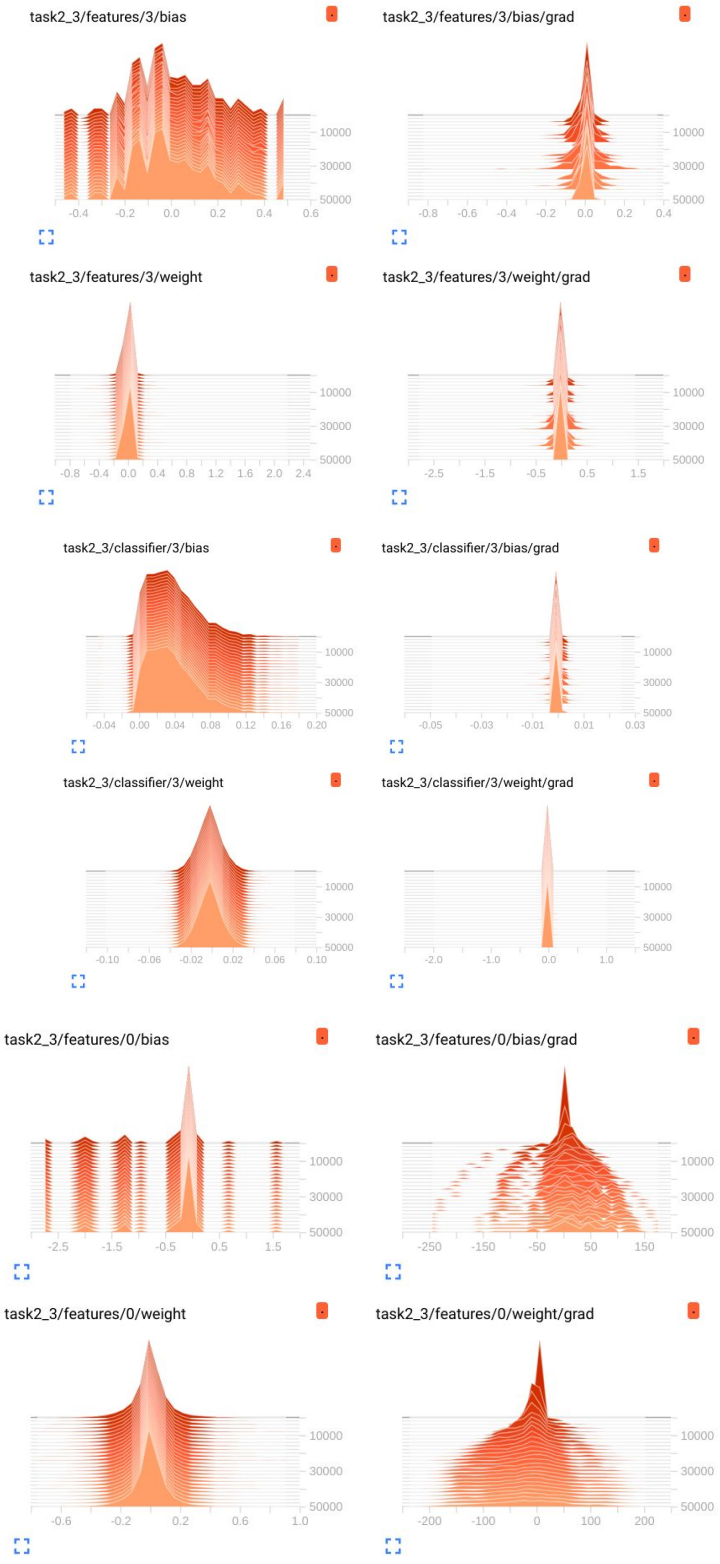
2.4.1 visdom downloaded image of training loss vs iterations



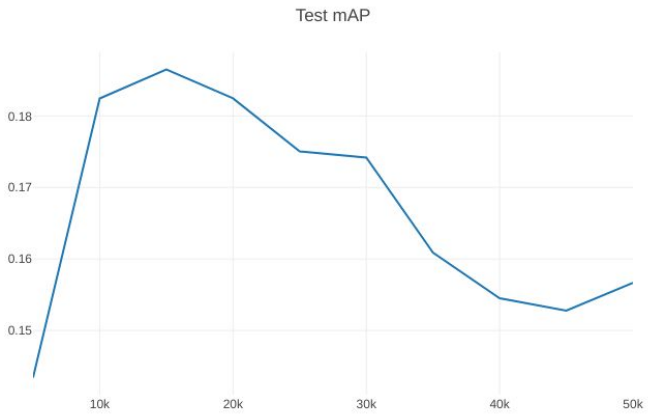
2.4.2 tensor board screenshot of training loss vs iterations



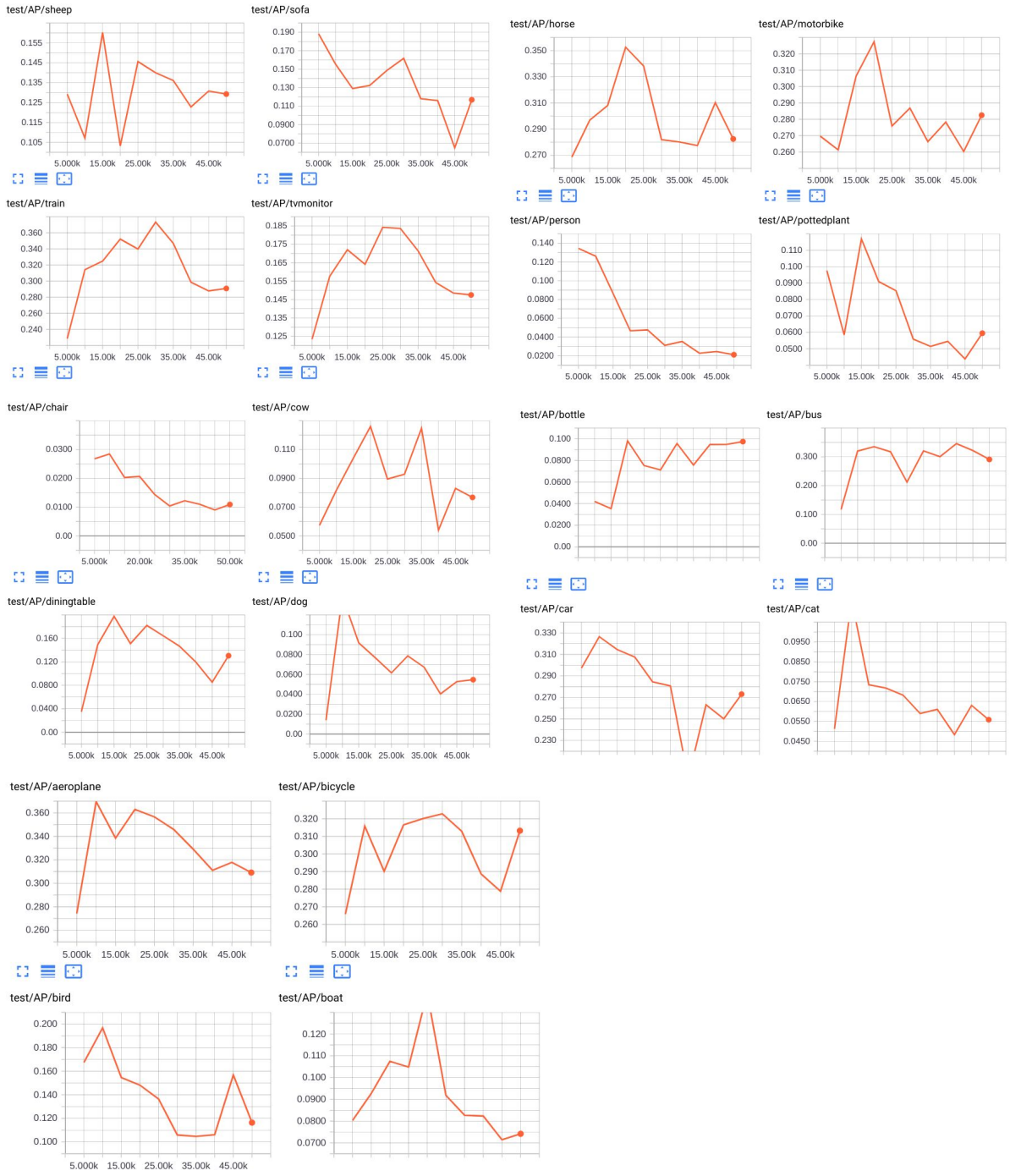
2.4.3 tensor board screenshot histogram of gradients of weights for conv1, conv2 and fc7



2.4.4 visdom downloaded image of test mAP vs iterations plot



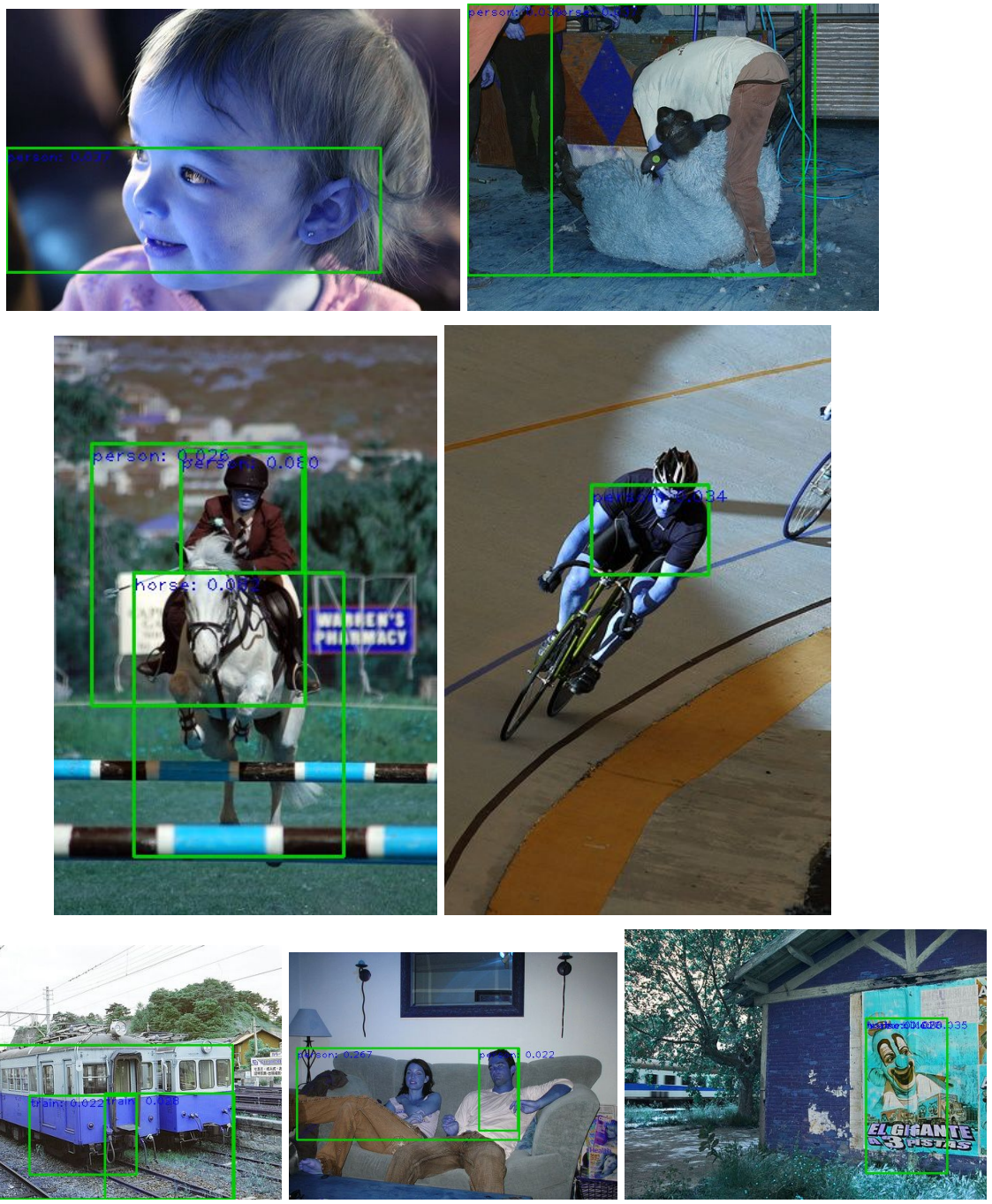
2.4.5 tensorboard screenshot for class-wise APs vs iterations showing 3 or more classes



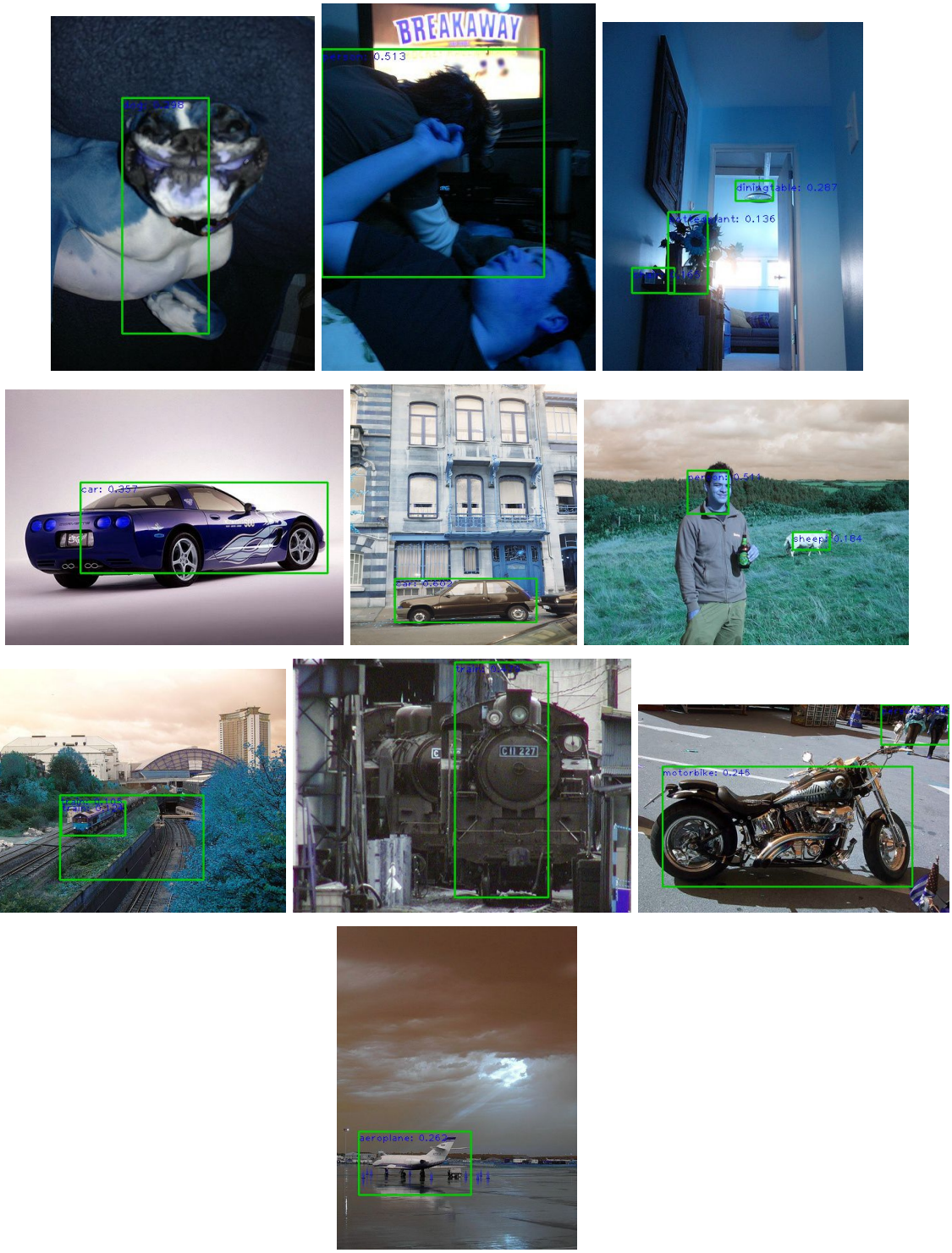
Observation:

Just like HW1, pottedplant and bottle account for the lowest mAPs. Automobiles like train, car, aeroplane, bike, motorbike had the highest mAPs. However, in many classes, mAPs decreased after some iterations. This indicates the model started to overfit the training data.

2.4.6 tensor board screenshot of images with predicted boxes for the first logged iteration (5000)



2.4.7 tensor board screenshot of images with predicted boxes for the last logged iteration (50000 or 45000)



2.4.8 report final classwise APs on the test set and mAP on the test set

AP for aeroplane = 0.3091
AP for bicycle = 0.3132
AP for bird = 0.1164
AP for boat = 0.0741
AP for bottle = 0.0974
AP for bus = 0.2909
AP for car = 0.2730
AP for cat = 0.0559
AP for chair = 0.0109
AP for cow = 0.0767
AP for diningtable = 0.1304
AP for dog = 0.0547
AP for horse = 0.2825
AP for motorbike = 0.2825
AP for person = 0.0212
AP for pottedplant = 0.0595
AP for sheep = 0.1293
AP for sofa = 0.1169
AP for train = 0.2909
AP for tvmonitor = 0.1475
Mean AP = 0.1567